VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Hra Halali pro platformu Android a JavaSE

Game Halali for Android Platform and JavaSE

2015 Petr Kolek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Kolek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Hra Halali pro platformu Android a JavaSE
Game Halali for Android Platform and JavaSE

Zásady pro vypracování:

Cílem práce je vytvořit elektronickou verzi hry Halali, kterou bude možno spustit jak na platformě Android tak na platformě JavaSE na běžném počítači. Hra bude navržena tak, aby umožňovala hru více hráčů díky připojení k serveru.

Hra umožní:
1. Hraní hry více hráčů v prostředí internetu.
2. Využití hráče s jednoduchou umělou inteligencí.
3. Samostatnou hru bez připojení k serveru.
4. Připojení do současně vyvíjeného portálu her v jazyce Java.

Práce bude obsahovat:
1. Popis použitých technologií.
2. Implementaci výše popsaného programu (serverové části, klienta pro platformu Android a JavaSE).
3. Popis řešení s využitím diagramů jazyka UML.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9. Dostupné z: http://it-ebooks.info/book/2249/
[3] Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 16. dubna 2015 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, zejména panu Ing. Davidu Ježkovi, Ph.D. za jeho cenné rady při vývoji hry a dále mé rodině za poskytnutí podmínek pro vznik této práce.

Abstrakt

Cílem této bakalářské práce je vytvořit elektronickou verzi deskové hry Halali, která bude moci být spuštěna na zařízení s operačním systémem Android a také na platformě JavaSE, která poběží na klasickém počítači. Práci jsem rozdělil do čtyř částí. V první části se věnuji pravidlům hry, ve druhé části je popsána uživatelská příručka, ve třetí části se věnuji používaným technologiím a ve čtvrté části samotné implementaci hry.

Klíčová slova: Java, Android, halali, LWJGL, Libgdx, člověk, zvíře

Abstract

The aim of this thesis is to create an electronic version of the board game Halali, which will be run on devices running Android and JavaSE platform, that will run on a classical computer. My thesis is divided into four parts. The first part describes to the rules of the game, in the second part is a users guide, the third part narrates about used technology, and the fourth part is devoted to the actual implementation of the game.

Keywords: Java, Android, halali, LWJGL, Libgdx, human, animal

Seznam použitých zkratek a symbolů

JVM	–	Java Virtual Machine
WORA	–	Write Once, Run Anywhere
GPU	–	Graphic Processing Unit
XML	–	Extensible Markup Language
LWJGL	–	LightWeight Java Game Library
API	–	Application Programming Interface
JSON	–	Hyper Text Markup Language
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
GUI	–	Graphical User Interface
SDK	–	Software Development Kit
GUI	–	Android Virtual Device

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této bakalářské práce je vytvořit původní deskovou hru v elektronické podobě. Tato hra běží na všech operačních systémech, které mají nainstalovaný JVM a na platformě Android.
Praktická část je rozdělena do tří částí a výsledkem je spustitelná aplikace na platformě Java SE a na platformě Android. Dále pak server, na který se mohou hráči připojit v režimu hry více hráčů.
V první části je implementace režimu hry single player, tedy hra jednoho hráče proti umělé inteligenci a implementace multi player (hry dvou hráčů) v síti a na internetu. Hra jednoho hráče obsahuje vytvoření všech potřebných tříd a funkcí, tak aby se daly napojit do GUI. Což zahrnuje vytvoření hráčů, karet a implementaci všech potřebných funkcí. Po spuštění hry jednoho hráče, bude v GUI zobrazena herní deska s vygenerovanými kartami, informace o hráčích včetně jejich skóre a kdo je na řadě. Pro vytvoření hry dvou hráčů je nutné vyplnit IP adresu serveru na který se chce hráč připojit. Dále musí uživatel zadat název místnosti (room), kterou chce vytvořit nebo do které se chce připojit a musí vyplnit své jméno (přezdívku). Hra dvou hráčů probíhá stejně jako hra jednoho hráče s tím, že není připojena umělá inteligence a je vyžadováno připojení do sítě internet.
Ve druhé části je vytvoření internetového serveru pro hru více hráčů. Server podporuje více her najednou, díky vytvoření tzv. místnosti pro každou hru. Pro připojení na server je nutné poslat IP adresu, název místnosti (room) a jméno (přezdívku) hráče.
A třetí část se zabývá přizpůsobením na platformu Android. Prostředí na platformě Android je totožné s prostředím pro Desktop.
Textová práce je potom rozdělena do čtyř hlavních kapitol. V první kapitole se budeme zabývat vlastními pravidly hry. V pravidlech je napsáno, jak probíhá hra od začátku do konce, jaká jsou pravidla lovu a podobně. Ve druhé kapitole je uživatelská príručka, která zahrnuje, jak má uživatel aplikaci nainstalovat až po to, jak ji používat. Třetí kapitola se věnuje použitým technologiím a zobrazuje na některých jednoduchých příkladech, jak je implementovat v jazyce Java. A čtvrtá kapitola se již věnuje samotné implementaci hry včetně popisu grafického rozhraní aplikací.

2 Pravidla hry

2.1 Myšlenka hry

Hra zdánlivě o zvířátkách, ale ve skutečnosti je tato hra taktická a logická. Je určena pro dva hráče a trvá přibližně 40 minut. Dolní hranice věku by měla být 8 let a horní hranice není nikterak omezena a určitě se u ní nebudou nudit ani dospělí. Hra z roku 2000 jejímž autorem původní deskové hry je Rudi Hoffmann a ilustrátorem je Franz Vohwinkel. V této hře platí, že i myslivec se může stát kořistí! Jeden hráč totiž bojuje za myslivce a dřevorubce, druhý se zase ocitne v kůži zvířat - lišek a medvědů. Oba loví bažanty a kachny, a také sebe navzájem. Nejprve se pod zakrytými kartičkami rozkládá tichý les s posedem ve středu. Čím více kartiček je během hry otočeno, tím je hon divočejší a napínavější. Dřevorubec káci myslivci stromy, aby měl výhled a možnosti střílet. Na myslivce ovšem čeká medvěd, který má mohutné a silné tlapy a může jej ohrozit. Na začátku se hra může zdát nevyrovnána, ale v průběhu hry se vše může otočit. Čím déle se hra hraje, tím více se naskýtá příležitostí a možností pro taktiku.

2.2 Cíl hry

Cílem hry pro myslivce a dřevorubce je ulovit co nejvíce kartiček zvířat a naopak. Po otočení všech kartiček, zbývá posledních 5 tahů každému hráči k tomu, aby mohl zachránit co nejvíce svých kartiček pomocí 4 východů z lesa. Pak následuje výměna rolí hráčů a hru si zopakují. Vyhrává ten, kdo má nejvyšší skóre na konci hry.

2.3 Počet kartiček

- 8 kartiček s modrým pozadím
  - 2 medvědi
  - 6 lišek
- 10 kartiček s hnědým pozadím
  - 2 dřevorubci
  - 8 myslivců
- 30 neutrálních kartiček se zeleným pozadím
  - 7 kachen
  - 8 bažantů
  - 15 stromů jehličnatých a listnatých

Tyto kartičky si vždy hra vygeneruje sama.

2.4 Průběh hry

Začíná hráč, který hraje za myslivce a dřevorubce. V dalších tazích se hráči navzájem střídají. Hráč, který je na tahu má dvě možnosti.

1. Otočit kartičku kliknutím na ještě neotočenou kartičku. Otočit kartičku zpět již nelze.
2. Udělat pohyb kartičkou, který se provádí kliknutím na kartičku, se kterou chce hráč pohnout a kliknutím na místo na hrací ploše na něž chce kartičku přesunout. Pokud je tento pohyb možný provede se. Pokud pohyb možný není, hráč musí udělat jinou akci na hrací ploše. Hráč může hýbat pouze se svými kartičkami. Hnědými kartičkami hráč, který hraje za dřevorubce a lidi. Modrými kartičkami hráč, který hraje za medvědy a lišky. Kachnami a bažanty mohou hýbat oba hráči. Stromy zůstávají na svém místě, pokud je dřevorubec neskácí. Kartičkami lze pohybovat pouze rovně, nelze je přemístit uhlopříčně a pouze přes prázdná pole.
3. **Pokud hráč neudělá pohyb kartičkou do 20 vteřin, tak je na řadě druhý hráč**

2.5 Pravidla pro kartičky

- Medvěd se může posunout pouze o jedno pole a loví myslivce a dřevorubce
- Dřevorubec se také může posunout pouze o jedno pole a může kácet pouze stromy
- Myslivec se může hýbat o libovolný počet polí a loví všechna zvířata
- Liška se může také hýbat o libovolný počet polí a loví všechna zvířata
- Bažant a kachna se mohou hýbat o libovolný počet polí a neloví nikoho
- Stromy se nesmějí hýbat

2.6 Výměna stran

Pokud jsou všechny kartičky otočeny, pak má každý hráč 5 tahů na záchranu svých kartiček. Kartičky lze zachránit pouze 4 východy, které jsou nakresleny vždy uprostřed každé strany hracího pole. Na kartičku stačí pouze kliknout, a pokud je možno ji zachránit, odebere se ze hry sama. Následuje výměna stran, protože hra nemusí být vždy vyvážená. Následuje ten samý průběh hry.

2.7 Konec hry

Hra končí, pokud proběhla výměna stran, všechny kartičky jsou otočeny a uběhlo 5 kol, které mají hráči na záchranu svých kartiček. Vyhrává ten hráč, který má vyšší skóre na konci hry.

3 Uživatelská příručka

Tato příručka je určena pro uživatele, kteří mají chuť a zájem si zahrát hru halali na platformě Desktop nebo na mobilním telefonu. Pro spuštění na počítači budeme potřebovat nainstalovaný balíček Java SE a alespoň 120MB operační paměti pro hru. Ke spuštění na mobilním telefonu je potřeba operační systém Android, ideálně s rozlišením displeje 480 x 800 pixelů (není však podmínkou). Pro hru typu multi player je zároveň nutné mít z daného zařízení přístup k internetu.

3.1 Instalace hry

- Pro instalaci hry na stolním počítači stačí stáhnout hru a soubor HalaliGame.jar. Zkopírovat jej do námi zvoleného adresáře. Z tohoto adresáře se vždy bude hra spouštět.
- Pro instalaci na mobilní telefon je potřeba stáhnout hru a soubor HalaliGame.apk. Nyní je nutné připojit mobilní telefon pomocí USB kabelu k počítači. Po připojení telefonu k počítači, musíte povolit v nastavení systému přístup k velkokapacitnímu zařízení. Nyní se v počítači zobrazí nový disk (Váš mobilní telefon). Zvolte adresář, do kterého chcete nakopírovat hru HalaliGame.apk pro instalaci a hru do tohoto adresáře zkopírujte. Nyní stačí na hru kliknout, a při dotazu zda-li se má aplikace nainstalovat kliknout na 'Ano'. Po pár vteřinách se aplikace nainstaluje a je umístěna v aplikacích.

3.2 Spuštění hry

Spuštění hry na platformě Desktop i Android je jednoduché.

- Pokud je Java správně nainstalována a máte připojení k síti s přístupem k internetu, stačí na ikonku hry dvakrát poklikat levým tlačítkem myši. Druhá možnost spuštění je pomocí příkazového řádku. Pro spuštění z příkazového řádku musíte zapnout příkazový řádek a musíte se dostat do adresáře se souborem HalaliGame.jar. Nyní stačí napsat příkaz java -jar HalaliGame.jar a hra se spustí. Po spuštění hry se zobrazí menu hry.
- Pokud je hra nainstalována je potřeba připojit se k síti s přístupem k internetu a najít hru v seznamu nainstalovaných aplikací. Nyní klikněte na ikonku aplikace. Následně se spustí aplikace a zobrazí se menu hry. Snažte se mít co nejlepší připojení k internetu, protože se může stát, že vypadnete ze hry.

3.3 Ovládání hry

- Na platformě Desktop je k ovládání potřeba pouze myš či touchpad.
- Ovládání na platformě Andoid je výhradně pomocí dotykové obrazovky. Tlačítkem Volume Up a Volume Down můžete regulovat míru hlasitosti aplikace.

Po zobrazení menu máte 3 možnosti, buď to kliknete na tlačítko Single Player, Connect nebo End.
V případě volby tlačítka Single Player se po několika vteřinách spustí hra jednoho hráče s umělou inteligencí. To znamená, že proti Vám bude hrát počítač a ne druhý člověk.
Ovládání je jednoduché a je stejné jak na platformě Desktop, tak i na platformě Android, pro otočení kartičky stačí kliknout na Vámi vybranou ještě neotočenou kartičku. Pro akci kdy například liška uloví bažanta stačí kliknout na lišku, její kartička se označí žlutým rámečkem a na kartičku bažanta. Následně se provede přesun kartiček a hráči se přičtou body. V této hře budete začínat Vy a budete tedy první klikat na kartičku. Pokud neprovedete žádnou akci do 20 vteřin následuje tah umělé inteligence. V momentě, kdy otočíte všechny kartičky a provedete posledních 5 kol do konce hry, můžete během těchto pěti kol se svými kartičkami odejít z lesa kliknutím na danou kartičku (kartička ovšem musí být v řádku či sloupci s východem z lesa a musí mít možnost pohybu), pak následuje výměna stran. Výměna stran trvá 5 sekund a je zobrazen nápisem Reverse Side. Po odehrání i druhého kola hry, tedy po výměně stran je konec hry. Zobrazí se Vám výsledek, kde uvidíte kdo vyhrál. Poté můžete zpět do menu (tlačítko Back To Menu), kde si můžete zahrát znovu, a nebo můžete ukončit hru tlačítkem End Game. Hra se dále řídí běžnými pravidly hry a ty jsou shrnuta v tabulce č.1.

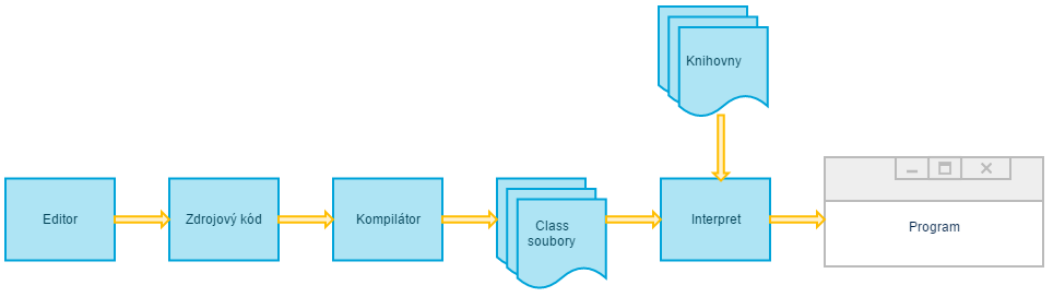
Karta	Počet	Body	Posun	Směr útoku	Cíl útoku
medvěd	2x	10	o 1 pole	do všech stran	myslivec, dřevorubec
liška	6x	5	libovolný	do všech stran	bažant, kachna
dřevorubec	2x	5	o 1 pole	do všech stran	oba druhy stromů
myslivec	8x	5	libovolný	ve směru hlavně pušky	medvěd, liška, kachna, bažant
bažant	8x	3	libovolný	žádný	žádný
kachna	7x	2	libovolný	žádný	žádný
stromy	15x	2	žádný	žádný	žádný

Tabulka 1: Shrnutá pravidla pro kartičky hry

4 Použité technologie

4.1 Java

Java je objektově orientovaný jazyk, začala být vyvíjená v roce 1991 týmem lidí, ve kterém byl zakladatel James Gosling a kolegové Patrick Naughton, Chris Warth, Ed Frank a Mike Sheridan ze společnosti Sun Microsystems. Původní jméno jazyka se v roce 1995 při uvedení na trh změnilo z Oak na Java. Hlavní motivací k vytvoření tohoto jazyka Java byla potřeba nového jazyka, který by byl nezávislý na platformě. Software, který by se vyvinul měl sloužit zejména pro spotřební elektroniku, jako jsou topinkovače, mikrovlnné trouby a dálkové ovladače. V době, kdy se Java začala vyvíjet nebyl takovýto jazyk k dispozici, jelikož všechny jazyky (např. C, C++) měly vytvořené kompilátory na určité použití. Vytváření těchto kompilátorů bylo navíc zdlouhavé a hlavně drahé.
Java se začala postupně rozšiřovat a stávala se populárnější díky tomu, že prohlížeče implementovaly jazyk Java ve formě appletů. V roce 1998 vyšla druhá verze Javy označována jako 1.2 a to hned v několika verzích. Java Standard Edition, která sloužila pro vývoj běžných aplikací, Java Enterprise Edition, která byla vhodná pro velké podnikové aplikace a Java Mobile Edition pro vývoj na mobilních platformách, protože byla zbavena některých funkcí. Javu postupně rozšiřovaly další verze, až do její dnešní podoby.
Od roku 2006 byla Java uvolněná jako open-source řešení, kromě některých částí kódu na které firma Sun Microsystem neměla práva. V roce 2009 převzala firmu Sun Microsystem firma Oracle. V té době odešel z týmu vývojářů Javy její zakladatel James Gosling.
Aby vývojáři nemuseli začínat zcela od začátku, tak Java zdědila vlastnosti z jazyků C a C++. Z jazyka C přebral syntaxi a z jazyka C++ převzal objektový model. Řada programátorů již znala tyto jazyky, a proto pro ně nebyl problém se naučit další nový. Java dodržuje standarty WORA, a díky nim je možné spustit aplikace na všech systémech v nichž je nainstalován JVM.
Java je jednoduchý a intuitivní jazyk a proto, oproti jiným jazykům, vývoj v Javě nedělá problémy ani začínajícím programátorům. Java se stará sama o správu paměti, a proto nepotřebuje funkce pro alokaci a dealokaci paměti jako například jazyk C. Správu paměti obsluhuje Garbage Collector, který v paměti automaticky uvolňuje místo, pokud na něj v aplikaci již neexistují žádné reference.
Tím, že se jazyk zkompiluje do Java Byte kódu a následně se spouští v JVM, vzniká nevýhoda oproti jazyku C/C++, jelikož má vyšší datovou náročnost. Java je nevhodná pro psaní ovladačů a real-time aplikací. Průběh kompilace je zobrazen na obrázku č.1. K dnešnímu dni je aktuální verze Java 1.8 označována jako Java Standard Edition 8. Pro vývoj na zařízení s platformou android jsem použil JVM nazývaný Dalvik VM.
Další informace najdete v referencích [3, 2].



Obrázek 1: Průběh kompilace Javy

4.2 OpenGL

OpenGL [4] je multiplatformní open-source knihovna pro práci s grafikou. Podporuje 2D i 3D objekty. Komunikuje s GPU a provádí renderování objektů ve hrách.
OpenGL se začala vyvíjet v roce 1991 ve firmě Silicon Graphics Inc. a uvolněná byla v roce 1992. Do té doby nebylo vyvinuto řešení pro práci s 2D a 3D objekty. Všichni programátoři si museli napsat vždy svou vlastní knihovnu pro práci s grafikou. Jejím největším konkurentem je knihovna Direct3D od společnosti Microsoft, která je obsažena v balíčku DirectX.

4.3 LWJGL

LWJGL [5] je open-source Java knihovna s multiplatformní podporou ke přístupu k špatně dostupným a užitečným API pro vývoj, jako je grafika OpenGL, audio OpenAL a paralelním systémům OpenCL. Tento přístup je přímý, vysoce výkonný a přehledný pro vývojáře. Tuto knihovnu používá řada frameworků a enginů, a pro vývojáře je lehčí pracovat s enginem nebo frameworkem, než přímo s knihovnou LWJGL.
www.lwjgl.org

4.4 LibGdx

Libgdx [6] je multiplatformní herní a vizualizační framework s podporou 2D i 3D her, určený pro začínající vývojáře. V současné době je dostupný pro cílové platformy, jako je Windows, Mac OS X, Android, iOS a HTML5.
Podporuje možnost napsat kód jednou a spustit ho na jiných platformách bez nutnosti vytvářet zásadní změny v kódu. Vývojář může použít veškeré funkce a vlastnosti Javy. Libgdx dodává vývojáři přímý přístup k file systému, vstupní zařízením, audio zařízením a OpenGL prostřednictví OpenGL ES 2.0 a 3.0 rozhraní. Dále je k dispozici mnoho API, které pomáhají vývojáři vytvářet celou hru. Mezi některá API můžeme zahrnout například renderování grafiky a textu, vytváření uživatelského rozhraní, přehrávaní zvukových efektů, streamování hudby, zpracování JSON a XML souborů a mnoho dalšího.

Díky tomu, že se všechny funkce skrývají za Java API, tak se vývojář nemusí bát, že by se při kompilaci pro některé platformy mohly vyskytnout chyby. Většinu těchto chyb již obešla Libgdx platforma a nemusí se s nimi vypořádat vývojář.

```
public class MyGame extends Game{

    MainMenuScreen mainMenuScreen;

    @Override
    public void create(){
        mainMenuScreen = new MainMenuScreen(this);
        setScreen(mainMenuScreen);
    }
}
```

Výpis 1: Ukázka kódu vytvoření hry

Pomocí zdědění třídy `Game`, která implementuje `ApplicationListener`, můžeme vytvořit třídu, která bude začínajícím bodem celé naší hry. Tato třída obsahuje implementaci metody `create()`, která se provede po spuštění, metody `render()`, kde se generuje herní smyčka a metody `dispose()`, která je nutná pro uvolnění např. grafiky a textur z paměti. V metodě `create()` můžeme pomocí funkce `setScreen()` nastavit obrazovku, která se má zobrazit. Obrazovky zase musí implementovat třidu `Screen`, ve které jsou metody `render(), pause(), resume(), resize()` a mohou být ještě `hide()` a `show()`.

```
public class Splash implements Screen {

    @Override
    public void render(float delta) {
    }

    @Override
    public void resize(int width, int height) {
    }

    @Override
    public void show() {
    }

    @Override
    public void hide() {
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
```

```
        }

        @Override
        public void dispose() {
        }

    }
```

Výpis 2: Ukázka vytvoření obrazovky

4.4.1 Stage

Stage v kinhovně Libgdx slouží pro ukládání, zobrazování a práci s 2D objekty. Pomocí `Stage` můžeme vytvářet celé hry, programy, menu nebo i různé levely her. Stage si můžeme představit jako jeviště na kterém se pohybují herci. Stage obsahuje metodu `act()`, která na každý actor dovoluje provést nějakou akci v závislosti na čase. Zároveň Stage zpracovává vstupní události - například klik, podržení myši, zmáčknutí klávesy, atd. K nezávislosti na rozlišení obrazovky můžeme použít jeden z Viewportů. `FitViewport` nastaví rozlišení tak, aby se vešlo na displej s tím, že se zachová poměr stran. To znamená, že po stranách mohou vzniknout černé pruhy. `StretchViewport` roztáhne rozlišení na celý displej s tím, že poměr stran se nemusí zachovat. Rozdíl mezi `FitViewport` a `StretchViewport` je na obrázku č.2.

Obrázek 2: FitViewport vs. StretchViewport

4.4.2 Actor

Třída actor slouží pro vkládání objektů do Stage v Libgdx. Actor obsahuje metodu `draw()`, jejž vykreslí grafiku objektu, metody `setX` a `setY`, která nastaví umístění ve `Stage`. Ukázka použití `Stage` a `Actor` je v ukázce kódu č.3.

```
FitViewport viewport = new FitViewport(width, height);

Stage stage = new Stage(viewport);

Actor actor = new Actor();

stage.addActor(actor);
```

Výpis 3: Ukázka použití stage

4.5 Gradle

Gradle je nástroj k automatizaci, šířen pod Apache-licence 2.0, tedy volně šířitelnou licencí. Poslední verze je 2.3 uvolněná 16. února 2015 a stále se vyvíjí. Je napsán v jazyce Java a Groovy, což je jazyk podobný Javě, avšak více podobný například Pythonu, tedy skriptovacímu jazyku. Byl vyvinut pro účely potřeby vytvořit build, zpřístupnit deployment, vygenerovat si dokumentaci nebo si připravit release. Je určen pro multiplatformní vývoj aplikací a her. Je to Domanin Specific Language (DSL), který nám umožňuje popsat to, co chceme zautomatizovat. Největší síla Gradlu je v buildování. Přebírá to nejlepší z Antu a Mavenu, z Antu sílu a flexibilitu a z Mavenu dependency managment a pluginy. Díky jednotlivým taskům, které obsahuje můžeme vytvořit build pro desktop nebo build pro Android. Po vytvoření projektu v Libgdx je právě typu Gradle.
Na ukázce kódu č.4 je zobrazen Gradle build soubor a jeho struktura. Je uvedena zdrojová cesta ke třídám, které se budou kompilovat, jméno projektu, knihovny, které se mají připojit. Tento soubor však může obsahovat více specifikací. Detailnější informace jsou uvedeny v referencích [gradle, gradle2].

```
apply plugin: "java"

sourceCompatibility = 1.6
[compileJava, compileTestJava]*.options*.encoding = 'UTF-8'

sourceSets.main.java.srcDirs = [ "src/" ]

eclipse.project {
    name = appName + "-core"
}

dependencies {
    compile 'com.esotericsoftware:kryonet:2.22.0-RC1'
}
```

Výpis 4: Ukázka konfiguračního souboru gradle

4.6 Kryonet

Kryonet [7] je open-source knihovna sloužící pro vytvoření spojení mezi klientem a serverem, buď to pomocí TCP nebo UDP spojení. Díky možnosti volby mezi TCP nebo UDP, je možno použít Kryonet pro realtime aplikace i pro aplikace, které jsou zaměřené na kvalitu spojení. Tuto knihovnu lze použít pro platformu Desktop i Android.
Kryonet pro serializaci i deserializaci využívá svou serializační knihovnu Kryo. Kryo automaticky serializuje objekty jak do, tak i z bajtů. Umí serializovat a deserializovat například boolean, Boolean, byte, Byte, Byte[], char,Character, short, Short, long Long, float Float, atd...

4.6.1 Vlastní serializátor

Pokud chceme některou třídu serializovat pomocí našho vlastního serializátoru, tak tato třída musí obsahovat implementaci rozhraní `KryoSerializable` a konkrétně metody `read()` a `write()`. Vytvoření vlastního serializátoru je vyobrazeno na výpisu č.5.

```
public class MyClass implements KryoSerializable{
    public void write  (Kryo kryo,  Output output){

    }

    public void read(Kryo kryo, Input  input){

    }
}
```

Výpis 5: Ukázka vytvoření vlastního serializátoru

Pokud chceme vytvořit instanci serveru, musíme použít třídu `Server`. Server může získat objekt serializátoru Kryo pomocí metody `getKryo()`. A můžeme mu zaregistrovat všechny objekty, které budeme posílat, nebo můžeme pomocí příkazu `getKryo().setRegistrationRequired(false)` zakázat povinnost registrovat posílané objekty. Dále musíme nabindovat server na určitý port, na kterém bude server naslouchat a následně jej spustíme metodou `start()`.
Náš server musí implementovat metody z rozhraní `Listener`. Pomocí metody `addListener()` můžeme na serveru nastavit naslouchání. Dále implementujeme metody `received()`, která definuje chování, pokud na server dorazí nějaká data, metodu `connected()`, která definuje co se má stát pokud se k serveru připojí klient, metodu `disconected()`, která definuje co se má stát, když se od serveru klient odpojí nebo nastane chyba ve spojení mezi klientem a serverem a metodu `idle()`, která definuje chování při nečinnosti spojení. Ukázka kódu pro vytvoření serveru je zobrazen na obrázku č.6 a klienta na obrázku č.7.

```
public class MyServer extends Listener{
    static  Server server;
```

```
    static int udpPort = 27960, tcpPort = 27960;

    private void startServer() throws IOException{
            server = new Server();
            server.getKryo().setRegistrationRequired(false);
            server.bind(tcpPort, udpPort);
            server.start();
            server.addListener(new MyServer());
    }
}
```

Výpis 6: Ukázka vytvoření serveru

```
public class MyKlient{
    public void startClient() throws IOException{
        private Client client;

        private static int udpPort = 27960;
        private static int tcpPort = 27960;
        private String ip = "127.0.0.1";
        private static int TIMEOUT = 60000;

        client = new Client();
        client.getKryo().setRegistrationRequired(false);

        client.start();
        client.connect(TIMEOUT, ip, tcpPort, udpPort);

    }
}
```

Výpis 7: Ukázka vytvoření klienta

4.7 Android SDK

Android SDK [14] je balíček určený pro vývojáře aplikací, kteří chtějí vytvářet aplikace na platformě Android. Balíček je dostupný na všechny hlavní platformy operační systémů (Windows, Mac OS i Linux). Balíčky SDK obsahují různé konfigurace a dostupné nástroje. Záleží na tom jak velký balíček si stáhneme.

- SDK Tools - obsahuje nástroje na debug aplikací (ddms), testování aplikací a Android emulátor.
- USB Driver – Komponenta, která je nutná pouze při ladění a testování aplikace nainstalované na zařízení. Potřebné pouze pro platformu Windows.
- Dokumentace – Obsahuje lokální kopii dokumentace pro aktuální Android framework API.
- Příklady kódů – Obsahuje ukázkové kódy aplikací, které jsou aktuální pro každou platformu.

- Google API - Knihovny díky kterým může vývojář využít rozhraní Google Maps

Emulátor slouží pro testování aplikace bez nutnosti připojit fyzický hardware k počítači a je obsažen v Android SDK. Pomocí Android SDK a AVD Manager je možné vytvářet virtuální zařízení s podporou SD karet, různých velikostí operačních pamětí a pak takto vytvořená zařízení spouštět. Většina aplikací ve virtuálním zařízení se chová jako ve fyzickém zařízení, ovšem na některé vyjímky, které se hůře simulují, jako je například příjmání hovorů nebo úroveň nabité baterie.

4.8 XML

XML [13] je značkovací jazyk, který byl vivinut a standardizován konsorciem W3C a je zdarma přístupný všem uživatelům. Používá se pro serializaci dat, takže tedy soupeří s JSON či YAML. Řada programovacích nástrojů podporuje právě zpracování XML jazyka. Jazyk je určen zejména pro výměnu dat mezi aplikacemi, k publikování dokumentů kde popisuje jejich strukturu a nezabývá se vůbec vzhledem. Jazyk vznikl jako potřeba kvůli tomu aby se nemusel používat specialní software k otevírání různých formátů (DOC, XLS ...), protože ne každý operační systém může podporovat příslušný software. Na jazyku uživatele nezáleží, protože XML od začátku podporuje různé znakové sady.
Pro vyznačení jednotlivých části textu používáme tzv. tagy. Tagy jsou párové značky, které ohraničují text, či další tagy. Ukázka je vyobrazena na výpisu kódu č.8.

```
<?xml version="1.0" encoding="ISO−8859−1"?>
<CATALOG>
  <CAR>
    <BRAND>Ford</BRAND>
    <MODEL>Mustang</MODEL>
    <COLOR>Red</COLOR>
    <SPEED>240</SPEED>
    <PRICE>200000</PRICE>
    <YEAR>1985</YEAR>
  </CAR>
<CATALOG>
```

Výpis 8: Ukázka XML kódu

4.9 TexturePacker

TexturePacker [9] je multiplatformní editor určený k vytváření kolekcí grafiky například ke hrám. Umožnuje uložit jakýkoliv grafický soubor do jednoho obrázku, tím ušetří místo v paměti bez ztráty kvality na obraze.

4.10 JSON

JSON [10] je datový formát, tedy způsob zápisu dat , který je nezávislý na počítačové platformě. Je určen pro výměnu dat, která mohou být orgnizována do polí nebo i do objektů. Vstupem je libovolná datová struktura a výstupem je vždy řetězec. Ukázka, jak může vypadat JSON soubor ,je vyobrazena na obrázku č.3.

```
{
    "com.badlogic.gdx.graphics.g2d.BitmapFont": {
        "font": { "file": "comicSans.fnt" }
    },

    "com.badlogic.gdx.graphics.Color": {
        "white": { "r": 1, "g": 1, "b": 1, "a": 1 },
        "black": { "r": 0, "g": 0, "b": 0, "a": 1 }
    },

    "com.badlogic.gdx.scenes.scene2d.ui.ImageButton$ImageButtonStyle": {
        "singlePlayer": { "imageUp": "singlePlayerS" , "imageOver": "singlePlayer", "pressedOffsetY": 0 },
        "invite": { "imageUp": "inviteS" , "imageOver": "invite", "pressedOffsetY": 0 },
        "connect": { "imageUp": "connectS" , "imageOver": "connect", "pressedOffsetY": 0 },
        "end": { "imageUp": "endS" , "imageOver": "end", "pressedOffsetY": 0 },
        "end2": { "imageUp": "end2S" , "imageOver": "end2", "pressedOffsetY": 0 },
        "menu": { "imageUp": "menuS" , "imageOver": "menu", "pressedOffsetY": 0 },
        "yes": { "imageUp": "YesS" , "imageOver": "yes", "pressedOffsetY": 0 },
        "no": { "imageUp": "noS" , "imageOver": "no", "pressedOffsetY": 0 },
        "createServer": { "imageUp": "createServerS" , "imageOver": "createServer", "pressedOffsetY": 0 }
    },

    "com.badlogic.gdx.scenes.scene2d.ui.Label$LabelStyle": {
        "default": { "font": "font" , "fontColor": "white"},
        "black": { "font": "font", "fontColor": "black"}
    },

    "com.badlogic.gdx.scenes.scene2d.ui.TextField$TextFieldStyle": {
        "default": { "background": "textfield", "font": "font", "fontColor": "white", "cursor": "cursor" }
    }
}
```

Obrázek 3: Ukázka navržení UI v libgdx pomocí JSON

5 Praktická část

V následující části se budeme zabývat postupným vývojem aplikace, chybami se kterými jsem se setkal při vývoji a kroky, které jsem provedl při jejím vývoji. Samotné spustitelné aplikace jsou tedy 3 a jsou vytvořeny pro spuštění klienta na platformě Desktop s JVM, s Android zařízením a bat souborem spustitelným na desktopu. Popis postupného vývoje je rozdělen do popisu Desktop klienta, síťové komunikace, mobilního klienta, návrhu a vytvoření GUI a problémů během implementace hry.

5.1 Požadavky na hru

V zadání mé bakalářské práce bylo, že hra má běžet na platformě Desktop a Android. Nejvíce vhodný framework byl Libgdx, jelikož podporuje možnost vytvořit build na platformu Desktop, Android, iOS HTML5. Má výborně popsanou dokumentaci k jazyku, přehledné tutoriály a pro mě lepší zpracování 2D hry oproti například JMonkey. Hra má podporovat hru jednoho hráče s umělou inteligencí a hru po síti. Ke hře po síti jsem si zvolil komunikaci klient - server a použil jsem knihovnu Kryonet, která usnadnila vytvoření serveru a klienta. Díky její propracované dokumentaci a podpoře i na platformě Android se vývoj aplikace urychlil. Komunikace klient - server znamená, že se uživatel připojí vždy na server pod jim zvoleným jménem a vyčkává na připojení druhého uživatele. O předávání jednotlivých paketů se stará tedy server.
Jelikož jsem si zvolil Libgx framework k implementaci hry, tak po vytvoření a otevření projektu mám pěkné rozdělený projekt v Gradle. Projekt byl rozdělen na `desktop`, `android` a `core`. Všechna implementace hry je vytvořena v core, v desktop a android je pouze implementován Launcher, který se stará o spouštění aplikace. Takže tedy jediný rozdílný kód mezi aplikací pro `desktop` a `android` je pouze ve v těchto vdou třídách. O kompilaci a build se stará Gradle, který pomocí svých příkazů provádí vybrané úkony.

Implementace hry v `core`, tedy její třídy byly pro přehlednost a nezávislost rozděleny do samostatných package, ty jsou vypsány níže.

com.halali.cards Obsahuje všechny třídy typu Card.

com.halali.game Obsahuje třídu pro spouštění aplikace.

com.halali.gamedata Obsahuje konstanty pro hru.

com.halali.graphics Obsahuje objekty, které se vykreslují ve hře. Například herní deska.

com.halali.packets Obsahuje pakety, které jsou nutné pro síťovou komunikaci.

com.halali.player Obsahuje třídy určené pro vytváření hráčů a PlayerManagera.

com.halali.screens V tomto package se nachází všechny obrazovky hry.

com.halali.server Obsahuje implementaci serveru.

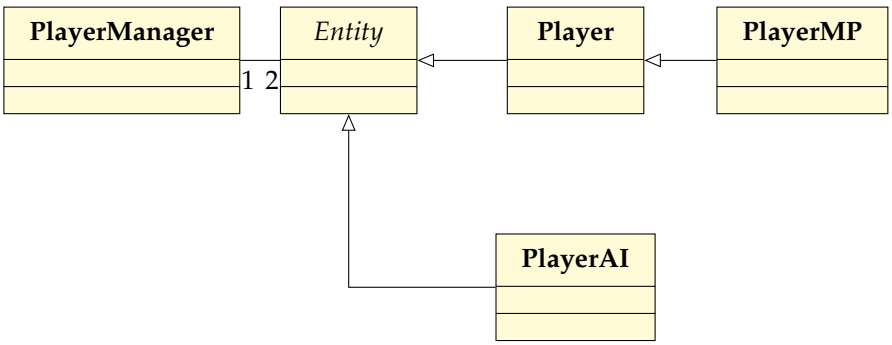
com.halali.sounds Obsahuje všechny zvuky ke hře.

5.2 Desktop klient

Klient pro platformu Desktop má pevnou šířku okna 760 pixelů a výšku 550 pixelů a nemožností toto okno maximalizovat nebo zvětšovat. Toto rozlišení jsem zvolil jelikož, všechna rozlišení na platformě `Desktop` dokáží zobrazit celé okno na běžném monitoru a tlačítka, včetně textu a kartiček jsou skvěle čitelná.

5.2.1 PlayerManager

Třída `PlayerManager` obsahuje vždy informace o dvou hráčích, kteří hrají proti sobě. Hráči jsou uložení jako `Entity`, jelikož můžeme přidat hráče `Player` nebo `PlayerAI`. Oba druhy hráčů jsou potomci právě třídy `Entity`. Mohou hrát buď to 2 hráči (`Player`) pro multiplayer, nebo jeden hráč (`Player`) a naprogramovaný hráč (`PlayerAI`) pro singleplayer. `PlayerManager` se stará o přičítání skóre, vypisování aktualního skóre, vypisování jmen hráčů, zobrazení daného hráče na tahu, střídání hráčů včetně 20 sekundového limitu. Během 20 sekund musí hráč udělat tah, jinak PlayerManager přepne hráče na tahu. V třídním diagramu na obrázku č.4 je zobrazena struktura package `Player`.



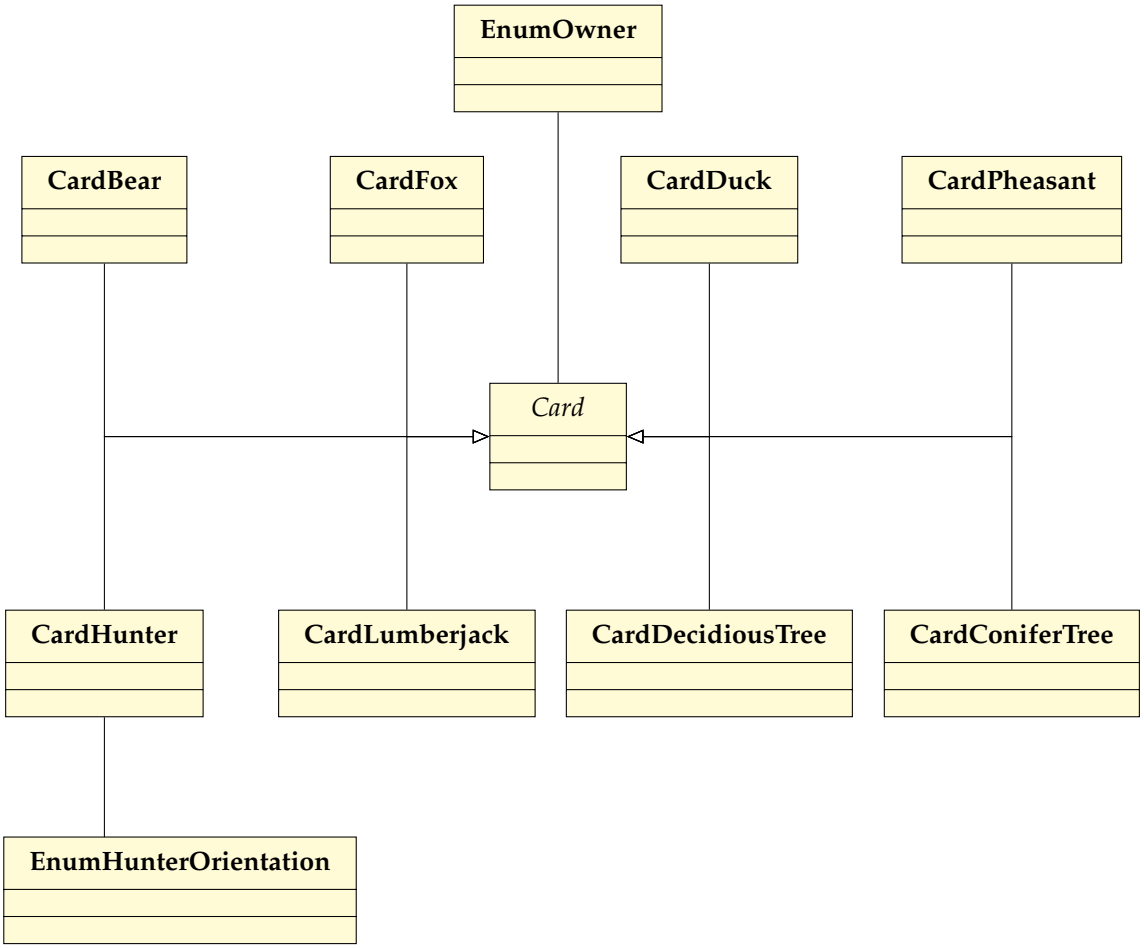
Obrázek 4: Class diagram třídy PlayerManager

Třída `Player` obsahuje informace o daném hráči, například jméno hráče, jeho skóre, zda je na řadě, zda je člověk, atd. Dále metody pro pohyb hráče, přičítání skóre, střídání hráčů, atd.
Třída `PlayerAI` je totožná s třídou `Player`. Liší se pouze logikou tahů. Jako u `Player` se vstup nebere z myši nebo dotykové obrazovky, ale provádí se na základě toho, zda je hráč na řadě. Pokud je hráč na řadě projde všechny otočené kartičky na desce a vybere si první možný tah, který může provést. Pokud nemůže provést žáden tah se svými kartičkami, potom otočí náhodnou kartičku na herní desce.

5.2.2 Card

Třída `Card` je abstraktní třída dědící z třídy `Actor`, ze které dědí ostatní karty jako například karta myslivce - `CardHunter`, dřevorubce - `CardLumberjack`, lišky - `CardFox`, atd. Každá karta obsahuje property `index, positionX, positionY, coordinateX`

`a coordinateY, isTurn, owner, isSelected` a `points`. Všechny karty také používají metodu `moveTo()` a `rotateCard()`, tedy kromě karty `SelectedCard`. Struktura karet je vyobrazena v třídní diagramu na obrázku č.5.



Obrázek 5: Class diagram package Cards

5.2.3 MouseAdapter

Třída `MouseAdapter` slouží pro zpracování odchytávání myši na obrazovce. Je použita zejména u hry jednoho hráče a u hry více hráčů. Díky této třídě můžeme pohybovat kartičkami a ovládat hru. `MouseAdapter` dědí ze třídy `InputAdapter`, která se nachází v balíčku knihovny Libgdx. Po vytvoření instance třídy `MouseAdapter` ji musíme nastavit do rozhraní `InputProcessor` pomocí metody `setInputProcessor()`.
Uvnitř této třídy přepisuji dvě metody ze třídy `InputAdapter`, a to `touchDown()` a `touchUp()`. Aby metoda `touchUp()` mohla být obsloužena, musí metoda `touchDown()` vrátit true. To znamená, že díky property `isHuman`, která je ve třídě `Player` můžu zajis-

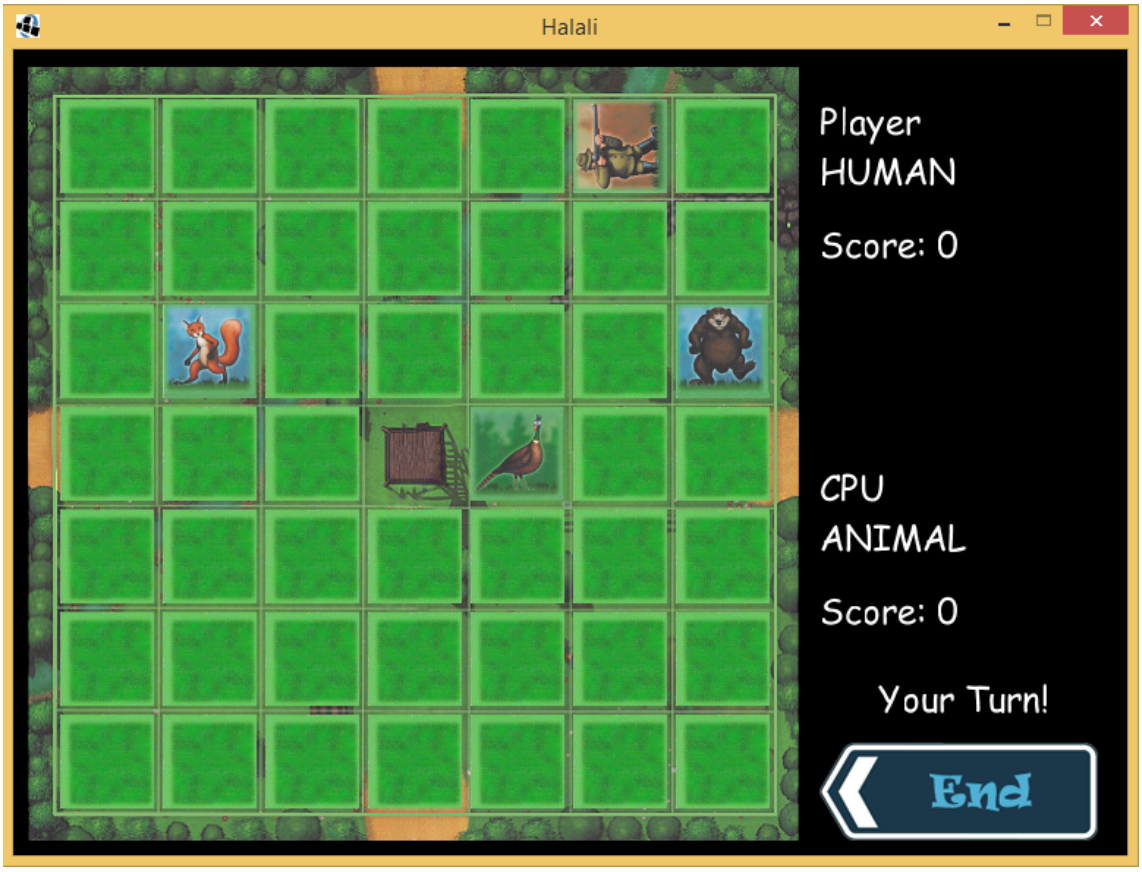
tit to, aby hráč nehrál pokud není na řadě.
Pokud tedy metoda `touchDown()` vrátí true, tak následuje metoda `touchUp()`, ve které se podle kliknutých souřadnic provede příslušná akce. Nejprve se zkontroluje zda nebylo kliknuto na nějaké tlačítko. Pokud nebylo kliknuto na tlačítko, zjistí se, na kterou kartičku bylo kliknuto pomocí toho, že projdeme pole všech karet a pomocí metody `isClick(x,y)` zjistíme zda bylo kliknuto na danou kartu. Pokud bylo kliknuto na kartu ukončí se procházení pole a vrátí se pozice karty v poli (ne souřadnice). Po kliknutí na kartu se ověří zda byla karta již otočena, pokud ne, otočí se, přičte se počet otočených karet a pokud ano označí se karta žlutým rámečkem pomocí funkce `selectCard()`, která příjmá jako parametr danou kartu. Jestliže již byla vybrána libovolná karta provede se pohyb je-li možný, pokud ne, karta se odznačí. Pokud jsou otočeny všechny karty, počítá se počet tahů do konce hry.

5.2.4 Hra jednoho hráče

Třída `ScreenGame` je obrazovka určená pro hru jednoho hráče a implementuje třídu `Screen`. Díky rozhraní `Screen` můžeme využít generování nekonečné herní smyčky. Je potřeba ovšem implementovat metody rozhraní `Screen`, jako je metoda `show()`, která se provede pouze jednou po spuštění rozhraní, dále metodu `render()`, která se neustále volá, metodu `resize()`, která zajišťuje správné zobrazení při změnšení okna, `pause()` se volá pokud hráč minimalizuje okno nebo má na zařízení s Android systémem příchozí hovor, nebo minimalizuje aplikaci, `hide()` metoda, která se zavolá pokud tato obrazovka již není aktuální pro hru.
Po vytvoření obrazovky se volá metoda `init()`, která se stará o vykreslení herní plochy, jmen hráčů, skóre a tlačítek. Když se herní plocha vykreslí, vygenerují se kartičky pomocí metody `initializeCards()` a přidají se na herní plochu. Poté začíná samotná hra. Ke hře je vytvořena metoda `selectCard(Card card)`, což je metoda pro výběr kartičky. Pokud daná kartička není otočena pak se otočí, jinak se pouze označí a čeká se na klik na druhé místo na herní ploše, vůči kterému chceme udělat tah. Pro odznačení kartičky slouží metoda `unselectCard(Card card)`. Metoda `reverseSide()` slouží pro změnu stran, pokud 1. část hry skončí. Pro ukončení celé hry slouží metoda `endGame()`, která se zavolá na konci hry nebo po kliku na tlačítko End Game. V metodě `endGame()` se zavolá nová obrazovka `ScreenResult`. Výsledná obrazovka hry jednoho hráče na platformě dekstop je zobrazena na obrázku č.6 a třídní diagram třídy `ScreenGame` je vyobrazen na obrázku č.7.

5.3 Síťová komunikace

Knihovna KryoNet umožnuje vytvořit TCP i UDP spojení mezi klientem a serverem. Jelikož mi nezáleží tolik na rychlosti komunikace, ale spíše na spolehlivém přenosu dat, proto jsem zvolil jako komunikační protokol TCP protokol.
Pro komunikaci mezi klientem a serverem jsou vytvořeny různé pakety, které využívají buď server i klient, a nebo pouze jeden z nich. Nejdůležitější paket je `PakcetGameState`, jelikož přenáší po každém tahu informace o tom, jak se změnila herní plocha, kolik kol

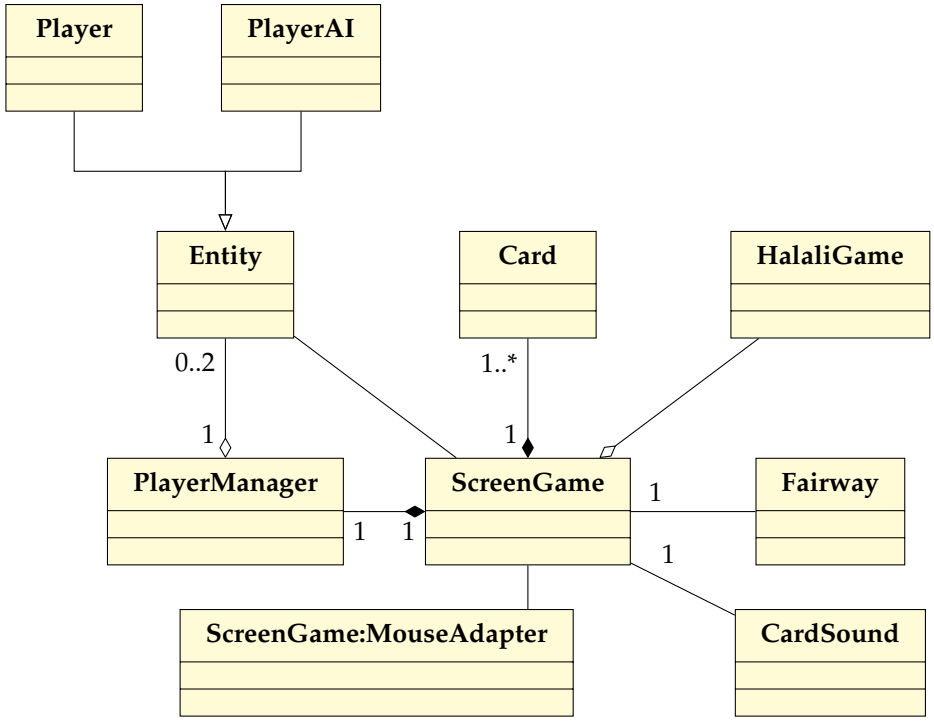


Obrázek 6: Výsledné GUI pro hru jednoho hráče

zbývá do konce hry, skóre, počet otočených karet a zda se mají karty inicializovat. Protože přenést celé pole objektů karet bylo velmi velké a tudíž ho nebylo možno přenést po síti, musely se objekty převést do pole `CardForPacket` pomocí funkce `setCards` a na přijmací straně jsou opět převedeny na třídu `Card` pomocí třídy `refreshCards`. Ve třídě `CardForPacket` jsou nejdůležitější informace pro přenos po síti. Používané pakety jsou zobrazeny v tabulce č.2 Jednotlivé pakety hry. Příklad jak může vypadat komunikace je na obrázku č.8 Ukázka síťové komunikace.
Pro mobilního klienta musel být upraven `manifest` přidáním permission pro ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE a INTERNET.

5.3.1 Herní server

Halali `Server` je herní server hry Halali určený pro hru více hráčů, který běží na platformě Desktop. Ke komunikaci mezi klienty a serverem je použita knihovna KryoNet. Server je vytvořen jako dávkový soubor bat. Vzhledem k tomu, aby se nemusel spuštět pro každou hru herní server zvlášť, je navržen tak, aby po spuštění vyčkával na připojení



Obrázek 7: Class diagram ScreenGame

klientů. Klientů se může připojit více a zároveň může na serveru běžet více her najednou. Po připojení do hry se přidá do listu `gameRoom`, nová místnost třídy `GameRoom` ve které jsou dva hráči, a která je jim vyhrazena do doby po kterou hrají. Aby server byl přístupný ke hraní odkudkoliv, musí být spuštěn na počítači s přidělenou veřejnou IP adresou.

5.3.2 Herní klient

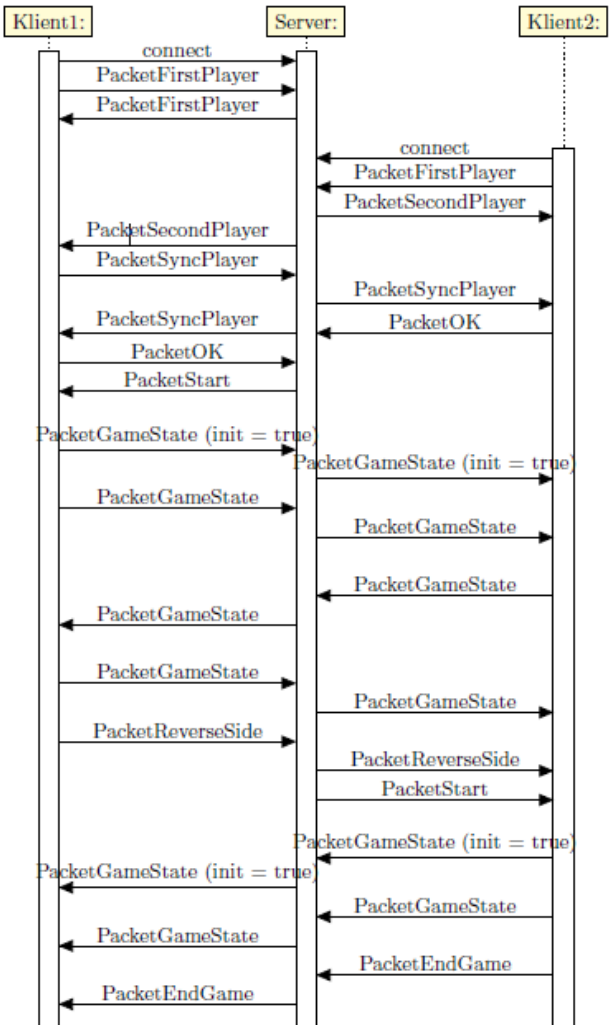
Třída `ScreenClient` je obrazovka, která implementuje rozhraní `Screen` a je určena pro hraní hry po síti. Je podobná obrazovce `ScreenGame` a obsahuje navíc metody pro navázání spojení se serverem tj. metodu `connect()`, metodu `received()`, která obsluhuje příchozí pakety, zpracovává je a podle nich provádí změny hry. Dále `ScreenClient` má metodu `createGameState()`, která vytvoří paket aktuálního stavu hry, který se má přeposlat pomocí funkce `sendGameState()` na server a z něj ke druhém klientovi.

5.4 Mobilní klient

Pro hru, která běží na mobilním telefonu jsem se rozhodl zachovat vzhled hry, jako tomu bylo na Desktop aplikaci. K tomu, aby se rozhraní na mobilním telefonu shodovalo s rozhraním na desktopu, jsem použil viewport, který je součástí knihovny Libgdx. Viewportů je v knihovně Libgdx více, já jsem si ovšem vybral `FitViewport`, který přijímá v konstruktoru výšku a šířku, kterou chci zobrazit, a který nezmění poměry stran. Tudíž

Typ paketu	Popis paketu	Odesíla server	Odesíla klient
PacketEndGame	Požadavek o ukončení hry	ANO	ANO
PacketFirstPlayer	Připojení první klienta	ANO	ANO
PacketGameState	Odeslání stavu hry	ANO	ANO
PacketLeaveGame	Odpojení klienta od serveru	ANO	ANO
PacketOK	Klient je připraven hrát	NE	ANO
PacketReverseSide	Požadavek o změnu stran a zahrání si druhé hry	ANO	ANO
PacketRoomExist	Odpoveď serveru, že daná místnost již existuje	ANO	NE
PacketSecondPlayer	Odeslání jména druhého klienta	ANO	NE
PacketStart	Zahájení hry	ANO	NE
PacketSyncPlayer	Zpráva pro klienty kdo je jejich protihráč	ANO	ANO

Tabulka 2: Jednotlivé pakety hry



Obrázek 8: Ukázka komunikace

se nestane, že by kartičky byly třeba obdélníkové atd. Kvůli tomu se ovšem na většině displejích zobrazí černé pruhy po stranách. Hra je orientovaná na mobilním zařízení na šířku displeje. Návrh rozhraní je vyobrazen na obrázku č.9.

5.4.1 AndroidLauncher

`AndroidLauncher` je třída sloužící ke spouštění Android aplikace na mobilním telefonu, která dědí z třídy `AndroidApplication`. Třída přepisuje dvě metody a to metodu `onCreate()` a metodu `onDestroy()`. V metodě `onCreate()` je ve funkci `initialize()` inicializace hry. Vytváří se `HalaliGame()`, to je třída v package com.halali.game, kde se zobrazí hlavní menu hry. Metoda `onDestroy()` obsahuje řádné ukončení aplikace a

Obrázek 9: Zobrazení GUI na mobilním telefonu

procesu, který tato třída vytvořila po spuštění aplikace. Ukázka třídy AndroidLauncher je vyobrazena ve výpisu kódu č.9.

```
public class AndroidLauncher extends AndroidApplication {
    @Override
    protected void onCreate (Bundle savedInstanceState) {

            super.onCreate(savedInstanceState);

            AndroidApplicationConfiguration config = new AndroidApplicationConfiguration();

             initialize  (new HalaliGame(), config);
```

```
    }

        @Override
        protected void onDestroy(){
            super.onDestroy();
            this. finish () ;
            Process.killProcess( Process.myPid() );
        };
}
```

Výpis 9: AndroidLauncher

5.4.2 Manifest

`Manifest` je konfigurační XML soubor pro Android aplikaci. V úvodu dokumentu specifikujeme verzi XML souboru a jeho kódování. V tagu `manifest` je zadán package, ze kterého je aplikace vytvářena, zároveň je udána verze aplikace. V tagu `uses-sdk` je udána minimální a cílová verze Anroid SDK. Následují tagy `uses-permission`, ve kterých jsou vypsána všechna potřebná povolení aplikace (přístup k síti, přístup k file systému nebo ke kameře). Název aplikace, cesta k ikoně aplikace, barevné schéma a další jsou obsaženy v tagu `application`. Mezi tagem `application` je tag `activity`, který slouží pro spuštění vybrané třídy po startu, jak má být zobrazena obrazovka, zda na šířku či na výšku, nebo zda se má skrýt, či zobrazit klávesnice na obrazovce.

5.5 Návrh GUI

Grafické rozhraní bylo navrženo pomocí grafického editoru Photoshop. Při návrhu GUI jsem se rozhodl vytvořit jednoduché a přehledné menu. Všechna tlačítka jsem vytvořil a uložil pomocí TexturePacker do jednoho obrázku, který se poté při načítání dané textury ořezává. Následně jsem si vytvořil konfigurační soubor `skin.json`, který obsahuje nastavení například pro tlačítka, font, barvy atd. Poté když jsem potřeboval v aplikaci použít například dané tlačítko stačilo zadat z jakého konfiguračního souboru se má tlačítko načíst a jak se jmenuje.
Samotná hra je po spuštění rozdělena na 3 zóny. První zónou je samotná hrací plocha, která byla naskenována spolu s kartičkami z původní deskové hry Halali a dále upravena na vhodné rozlišení nacházející se v levé části obrazovky. V druhé části se nachází informace o hře, jako jsou jména hráčů, role ve hře, jejich skóre, kdo je na řadě a tlačítko k ukončení hry. Třetí část je viditelná až po kliknutí na tlačítko End Game, ve které se nachází dotaz zda si je hráč jist, že chce hru ukončit a dvě tlačítka s možnostmi ano, ne.

5.6 Problémy se kterými jsem se setkal

5.6.1 Generování kartiček

Původně jsem generování kartiček udělal tak, že se vygenerovalo náhodné číslo a podle tohoto čísla se vytvořila instance dané kartičky do pole `cards[][]`, pokud ovšem již

nebyly vygenerovány všechny kartičky daného typu. Poté se musela vygenerovat jiná kartička. Zásadní problém byl v tom, že jsou různé počty kartiček a při vygenerování byla vždy velmi podobná herní plocha. A jeden typ karet se generoval pospolu. Proto jsem přistoupil na postup kdy si vygeneruji všechny kartičky postupně do listu, ten se poté dvakrát zamíchá a následně uloží do pole `cards[][]`.

5.6.2 Přerušování hlavního vlákna

Herní vlákno běží samostatně a pokud jsem potřeboval pustit například timer, pak i ten běžel ve svém vlastním vlákně, tímpádem ve vlákně timeru chyběl OpenGL kontext. Proto jsem si musel vytvořit `HelperThread`, který má property a podle nastavených property se vykoná daný kód. Řešením tedy bylo nastavit, co se má stát v HelperThread a spustit jej přes `Gdx.app.postRunnable(helperThread)`.

5.6.3 Přepočet souřadnic screen to stage

`Stage` i `InputAdapter` používají rozdílný výchozí bod pro počítání souřadnic. `Stage` používá levý dolní roh a `InputAdapter` používá levý horní roh. K tomu abych vyřešil tímto vuniklou chybu, bylo potřeba převést souřadnice z obrazovky na souřadnice `stage` zapomocí metody `screenToStageCoordinates()`. Tuto chybu jsem odhalil až při debugu, kdy jsem si nechal zobrazit ohraničení všech karet a souřadnic na které se kliklo.

5.6.4 Špatně načtené textury v Android aplikaci

Jelikož v Androidu je životnost textury vázána na aktuální `Activity`, po novém spuštění aplikace se spustí proces se stejným id aplikace, avšak textury dále nejsou platné, z tohoto důvodu jsou nečitelné. Tudíž je nutné po ukončení aplikace ukončit daný proces, nebo načíst znovu všechny textury. Ukázka kódu ukončení procesu v Android aplikaci je zobrazen na výpisu kódu č.10.

```
@Override
protected void onDestroy(){
        super.onDestroy();
        this. finish () ;
        Process.killProcess(  Process.myPid() );
};
```

Výpis 10: Ukončení procesu v Android

6 Závěr

Cílem mé bakalářské práce bylo navrhnout a implementovat elektronickou verzi deskové hry halali pro platformu desktop a mobilní platformu Android.
Celá hra byla navržena tak, aby bylo odděleno co nejvíce GUI od logiky hry. Všechny třídy byly rozděleny do balíčků, které jsou na sobě nějakým způsobem závislé. Pokud budu chtít ve hře něco upravit, postačí změním-li to v dané třídě.
K implementaci této hry bylo nutné zvolit správně technologie se kterými se dá pracovat, jak pod platformou Java, tak i aby fungovala s mobilní platformou Android. Rozhodoval jsem se mezi použitím JMonkeyEnginu a Libgdx frameworku. Vzhledem k tomu, že framework Libgdx je více zaměřen na 2D a spolupracuje bez obtíží s knihovnou Kryonet pro komunikaci, zvolil jsem si jej. Při vývoji jsem narazil na problém, kdy aplikace po nějaké době začala spotřebovávat více operační paměti než by měla, což bylo způsobeno nesprávným používáním metody dispose() a uvoňováním zdrojů.
Hra nyní běží pod platformou JavaSE a Android. Fungují oba dva módy hry, jako je hra jednoho hráče proti umělé inteligenci a hraní jednoho hráče proti druhému hráči v prostředí internetu. V budoucnu by se mohla rozšířit a zdokonalit umělá inteligence, neboť nyní je pouze základní. V Android GUI jsem nemusel dělat velké změny, jelikož jsem použil FitViewport, díky kterému je hra hratelná na různých rozlišeních.
Vzhledem k tomu, že Java herní server, který měl být vyvíjen souběžně s touto bakalářskou práci nebyl vyvinut, nemohl jsem jej do hry implementovat. Vývoj hry prověřil do hloubky mé zkušenosti a prohloubil mé dovednosti v oblasti programování v jazyce Java a v programování pro platformu Android. Díky tvorbě této bakálařské práce jsem využil sázení textu v Latexu a naučil jsem se lépe řešit problémy se kterými jsem se setkal.

7 Reference

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025

[2] DARWIN, Ian F. Java Cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9.
Dostupné z: http://it-ebooks.info/book/2249/

[3] Java (programming language). [online]. [cit. 2015-04-18]. Dostupné z: http://en.wikipedia.org/wiki/Java_(programming_language)

[4] OpenGL. [online]. [cit. 2015-04-18]. Dostupné z: http://en.wikipedia.org/wiki/OpenGL

[5] [5] LWJGL. [online]. [cit. 2015-04-18]. Dostupné z: http://lwjgl.org/wiki/index.php?title=About_LWJGL

[6] LibGdx. [online]. [cit. 2015-04-18]. Dostupné z: https://github.com/libgdx/libgdx/wiki

[7] TCP/UDP client/server library for Java. [online]. [cit. 2015-04-18]. Dostupné z: https://github.com/EsotericSoftware/kryonet

[8] LibGDX Tutorial series. [online]. [cit. 2015-04-18]. Dostupné z: http://www.gamefromscratch.com/page/LibGDX-Tutorial-series.aspx

[9] TexturePacker online documentation. [online]. [cit. 2015-04-18]. Dostupné z: https://www.codeandweb.com/texturepacker/documentation

[10] Introducing JSON. [online]. [cit. 2015-04-18]. Dostupné z: http://json.org/

[11] Gradle, moderní nástroj na automatizaci. [online]. [cit. 2015-04-18]. Dostupné z: http://www.zdrojak.cz/clanky/gradle-moderni-nastroj-na-automatizaci/

[12] Gradle. [online]. [cit. 2015-04-18]. Dostupné z: https://gradle.org/

[13] Extensible Markup Language. [online]. [cit. 2015-04-18]. Dostupné z: http://cs.wikipedia.org/wiki/Extensible_Markup_Language

[14] Android (operační systém). [online]. [cit. 2015-04-20]. Dostupné z: http://cs.wikipedia.org/wiki/Android_%28opera%C4%8Dn%C3%AD_syst%C3%A9m%29

A Obsah CD

Na přiloženém CD jsou následující soubory:

1. Spustitelná verze hry pro platformu Desktop a Android
2. Text této bakalářské práce ve formátu PDF
3. Zdrojové kódy ke hře